University of Cincinnati

Date: 6/9/2015

I, Rohit Bhoompally , hereby submit this original work as part of the requirements for the degree of Master of Science in Computer Science.

It is entitled:

Analysis of business ranking for a connected group of Yelp users by aggregating preference pairs

Student's name: **Rohit Bhoompally**

This work and its defense approved by:

Committee chair: Fred Annexstein, Ph.D.

Committee member: Raj Bhatnagar, Ph.D.

Committee member: Chia Han, Ph.D.

17302

Analysis of business ranking for a connected group of Yelp users by aggregating preference pairs

A thesis submitted to the Graduate School of **University of Cincinnati** in a partial fulfillment of requirements for the degree of **Master of Science** in the **College of Engineering and Applied Science** department

by

Rohit Bhoompally
Bachelor of Technology
Vardhaman College of Engineering, India
May 2011

Committee Chair: Prof. Fred Annexstein, Ph. D.

Abstract

The aim of this thesis is to analyze algorithms that efficiently rank businesses (restaurants, bars, theatres, cafes etc.) of a social network. In this thesis, we focus on the analysis of algorithms applied to a data set obtained from aggregated pair preferences. The algorithms we analyze include Zermelo's probability model, PageRank, and In-degree counting. We demonstrate significant differences in the results and performance of these diverse methods. Our results will ultimately allow us to provide more relevant and better recommendations to a social network user.

The main idea behind the work is perform preference pair aggregation to rank nodes of a business graph. A preference pair {A, B} for a user U suggests that - the user U prefers (better likes) business A to B. We use implicit feedback from the dataset to decide a user's preference. We try to aggregate such preference pairs for all businesses the user U and his friends F_1, F_2, F_3 … have reviewed to bring out meaningful recommendations for the user U. Once we have a large set of preferences pairs for a particular user, we run a Zermelo's probability model, PageRank and In-degree counting separately over it to rank the nodes, and compare the results.

In our experimental comparisons, we apply these methods to data obtained from the Yelp social network. We find that PageRank and In-degree perform better than Zermelo's model in aggregation with paired preferences. Moreover, we also see that the paired preference model performs superior to simple rating sort ranking. Also, through an analysis of the proposed model we find a few important insights for Yelp platform like - how similar/dissimilar are the preferences in a group of friends, and what percentage of top ranked businesses have already been visited by a user.

Table of contents

1 Introduction 7
1.1 Goal of research 7
1.2 Motivation 8
1.3 Approach and outcomes 9
2 Related work 13
3 Yelp dataset 18
4 Ranking businesses using preference pair aggregation 23
4.1 Preference pairs 23
4.2 Forming the graph 24
4.3 Assumptions and inconsistencies 26
4.3.1 Outliers 26
4.3.2 Business dissimilarities 27
4.3.3 Cycles for one user only graph 27
4.3.4 Cycles for a user circle graph 29
4.4 Batching 30
4.5 The idea 31
4.5.1 Get user circle 34
4.5.2 Get reviewed businesses 35
4.5.3 Get preference pairs 35
4.5.4 Social graph 37
4.5.5 Zermelo's probability model 37
4.5.6 In-degree counting 44
4.5.7 PageRank 44
5 Results and analysis 47
5.1 Experimental setup 47
5.2 Irreducibility 49
5.3 Recommendations via ranking businesses 51
5.4 Model comparison 53
5.5 User behavior in Yelp using commonality 64
6 Conclusion and future work 65

Table of figures

Figure 1: Nodes A and B with a directed edge in our social graph 11

Figure 2: MRRs for thumb-based and comparison-based ranking for α = 1.0. At m = 1000, for the best γ, MRR is 0.149 for comparison-based approach and 0.0186 for thumbs [10] .. 15

Figure 3: MRRs for thumb-based and comparison-based ranking for α = 1.5. At m = 1000, for the best γ, MRR is 0.244 for comparison-based approach and 0.019 for thumbs [10] 16

Figure 4: Overview of the cities involved in the Yelp Dataset [5]..... 19

Figure 5: Preference - indicating user U prefers business B1 to B2 24

Figure 6: A simple preference pair graph for a single user..... 25

Figure 7: Cycle not possible for a single user only graph with two businesses 28

Figure 8: Cycle not possible for a single user only graph with three businesses 29

Figure 9: An example of category tree for Yelp with two levels 31

Figure 10: Architecture for our ranking mechanism..... 33

Figure 11: Example of a win matrix M for three businesses 43

Figure 12: Example graph where A's in-degree is 1, B's in-degree is 3 and C's in-degree is 2.. 44

Figure 13: Preference pair JSON object for a user, where bi stands for business item..... 47

Figure 14: Sample CSV for a user who reviewed three businesses that Gephi understands ... 49

Figure 15: Social graph for a random user from Yelp dataset containing a disconnected hub generated using Gephi 50

Figure 16: Sample output of recommendations using both Zermelo and PageRank systems for one user..... 53

Figure 17: Few of many rank comparisons for different users..... 57

Figure 18: Mean Average Precision – Zermelo vs. PageRank vs. In-Degree (in aggregation with paired preference model)..... 58

Figure 19: Number of wins – Zermelo vs. PageRank vs. In-Degree (in aggregation with paired preference model) 59

Figure 20: Mean Average Precision – Paired preference model + In-Degree vs. Rating sort ... 60

Figure 21: Number of wins – Paired preference model + In-Degree vs. Rating sort..... 61

Figure 22: Mean Average Precision – Paired preference model + PageRank vs. Rating sort ... 62

Figure 23: Number of wins – Paired preference model + PageRank vs. Rating sort..... 63

1 Introduction

1.1 Goal of research

The goal of this research is to analyze algorithms that efficiently rank businesses in a social network (like Yelp) based on the user's activity (ratings and reviews of businesses he/she visited) in aggregation with his/her friend's activity. This ranking will ultimately provide us with a recommender system that will generate relevant and better recommendations of businesses. The basis of our model is paired comparisons of user's preferences about businesses. We do not ask the user for a timely feedback about which between two businesses, he/she prefers. Rather, we implicitly try to assume his preference using the reviews written by the user in the past. We will use three different ranking mechanisms - Zermelo's probability model, PageRank and In-degree counting, to rank our businesses on the social graph formed by aggregating the preference pairs. We compare these ranking mechanisms to see which one of these yields better results for our setup. We apply our model to an anonymized academic dataset exposed by Yelp [3] and rank the businesses based on popularity within the user's network and generate relevant suggestions.

1.2 Motivation

Recommendations, in the recent past, have been a highly concentrated area in the field of Computer Science. Almost every successful service presents recommendations to their users based on liking, preferences, context (time and location), activity, friend's activity and many other such factors. For example, Facebook suggests friends, YouTube suggests videos, Netflix suggests movies, Music streaming services like Rdio suggests music etc.

Yelp's recommendation system ranks reviews primarily. Other than recommending reviews, Yelp also suggests businesses. Yelp's website and the mobile apps have a varied set of suggestions in place. One of the popular sections in Yelp is "Best of Yelp: *City*". In this section, Yelp ranks businesses (food, bars, nightlife and so on) based on ratings and reviews given by Yelp community, and decide the most popular places to visit in a city. These suggestions are similar for all the users for that particular city. Yelp also suggests businesses based on location. In this thesis, we try to recommend businesses taking into account the user's network (in-network suggestions), which can potentially be a feature for Yelp.

A typical user, in a social network like Yelp, would want to know what new businesses have come up in recent times, and what businesses are emerging to be popular. One way to discover new and popular businesses is via the user's network. Being able to know businesses that a user's friends are visiting (via check-ins), and their liking towards a particular business (via ratings) and what exactly they like about the particular place, be it the service, hygiene, ambience etc. (via reviews) has become necessary in the modern world. In addition, in-network suggestions for a user can play a vital role in business discovery.

Therefore, we try to analyze different algorithms that efficiently rank businesses in a social network like Yelp. We apply these algorithms on a dataset obtained from aggregated preference pairs. Our results will ultimately provide us with better and relevant recommendations (in-network suggestions) of businesses in the social network.

1.3 Approach and outcomes

For the user of interest, we gather all of his friends from the Yelp dataset. Each friend will have his own activity related to the Yelp service, and he/she would have rated and reviewed a bunch of businesses. For every such friend, we gather all the businesses he/she reviewed, and we perform implicit paired comparison. What this means

is, we take each review written by the user and compare it with every other review written by the same user. We do not ask the user explicitly about what business among two businesses he likes more, rather we perform implicit comparison of businesses visited by the user (reviews written about businesses) using the rating and review text.

When comparing two businesses, we first look at the rating given by the user for both the businesses. Based on this value, we implicitly conclude that the user prefers the business that he/she has rated higher to the other one. If a user has rated both the businesses the same, we take the review text and perform sentiment analysis using Alchemy API [4]. This API returns a simple sentiment value between -1 and 1 indicating positive and negative sentiment. We conclude that the review for a business that has a value closer to 1 is more preferred than the other for a user. At this point, we will be able to compare any two businesses implicitly using review parameters (rating and review text) and decide the user's preference.

After we have gathered the preference pairs for each of the friends for our user of interest, we aggregate all the preferences into a social graph. Each node in the graph will represent a business, and each edge will represent a single user's preference. For example, a preference pair P {A, B} for a user U will be represented in our social graph as two

nodes A and B, with a directed edge between the nodes, as shown in Figure 1.

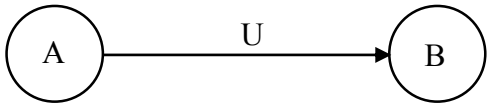

Figure 1: Nodes A and B with a directed edge in our social graph

The directed edge is an indication of preference. Simply put, the node that has an incoming edge is the one that is more preferred. Figure 1 indicates that user U prefers business B to business A.

Once we have our social graph, we rank the nodes of the graph to gather our recommendations. We try three different algorithms, Zermelo's probability model, PageRank, In-degree counting, to see which one of these perform better in aggregation with the paired preference model. We try to compare the three algorithms in aggregation with paired preference model using two metrics – MAP (Mean Average Precision), and number of wins (described in section 5.4), and see that PageRank and In-degree counting perform better in aggregation with paired preference model, with PageRank having a slight lead overall.

Furthermore, we compare the paired preference model in aggregation with the ranking mechanisms to the rating based ranking of businesses and find that paired preference model performs better. We also find some very interesting insights about the Yelp platform, like how similar/dissimilar are the preferences in a group of friends, and what percentage of top ranked businesses have already been visited by a user.

2 Related work

Peilin and Hui [9] in their paper investigate ranking-based strategy for contextual suggestions. They propose to rank candidate suggestions based on their similarity to the personal profile and that to the contexts (geographical and temporal information) over Yelp and foursquare datasets. Their ranking mechanism looks at the similarity between a suggestion and the places that the user like and the dissimilarity between a suggestion and the places the user dislikes based on the user's profile.

We take inspiration from Peilin and Hui [9] in drawing suggestions based on user-profiles. They try to present suggestions that are close to the locations that the users like, and try not to present those, which the user dislikes. In our thesis, we try to draw our suggestions from user profile – specifically from the friend's activity, that is, businesses visited by the friends of the user on Yelp. We then perform paired comparison to form a social graph, and run a ranking algorithm to rank the nodes of our graph.

Anish, Atish and Srinivas [10] in their paper show that comparison-based ranking mechanism performs better than thumb-based ranking schemes. Simply put thumb-based ranking approaches make use of a

quantity that signifies the importance of an item like star rating, thumbs up/ thumbs down count and so on, whereas comparison based ranking ranks items by comparing sets of two items with each other and update their scores. In their system, Shoutvelocity, when a user submits an item, he is presented with a pair of items drawn from the database for review. The database holds all the items at any time. Once the user reviews the items (that is likes one of the two presented items), the scores of the items are updated and stored to database. Current score of the items or the rank estimates bias the items that are picked for review and another factor that is considered while picking items for review is the age of the item, preferring newer items to older ones [10].

The algorithm for updating scores with comparison keeps track the number of times the items are compared, accounting for a discounting factor, which stabilizes the scores. Suppose the items i1 and i2 have been evaluated k1 and k2 times respectively. The discounting is done based on ci = 2 (1+ki) 0.5 where i = 1, 2. If the current score of items i1 and i2 is s1 and s2 and if i1 gets voted in the comparison, then s1 is incremented by c1 * (1− 1/(1+e^{s1-s2})) and s2 is decremented by c2 * 1/(1+e^{s1-s2}). Similarly, i2 is updated if it wins [10].

A comparison between thumb-based ranking approach and comparison-based ranking approach over 1000 candidates, where the items to be compared are picked based on their existing score (low scores items are given low preference) show that comparison based algorithm performs better. Their evaluation metric is MRR (mean reciprocal rank). MRR for an item is the mean of the quantity 1/R, where R is the rank of the actual top item produced by their approach. MMR is utmost 1 and MMR of 1 means the top item was correctly identified as the best item.

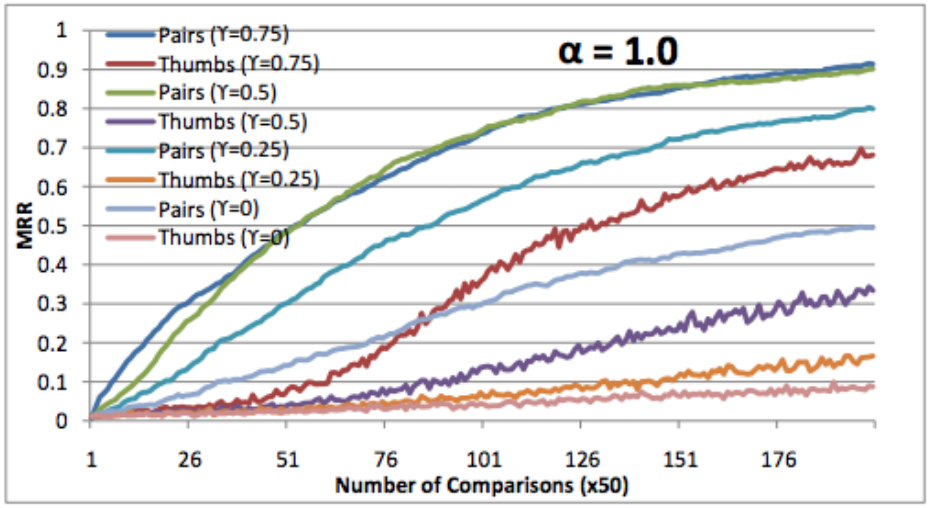

Figure 2: MRRs for thumb-based and comparison-based ranking for α = 1.0. At m = 1000, for the best γ, MRR is 0.149 for comparison-based approach and 0.0186 for thumbs [10]

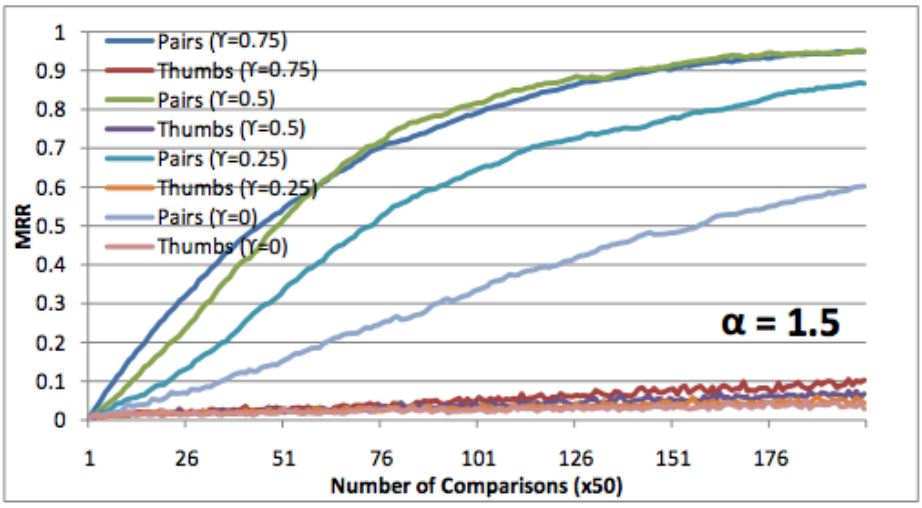

Figure 3: MRRs for thumb-based and comparison-based ranking for α = 1.5. At m = 1000, for the best γ, MRR is 0.244 for comparison-based approach and 0.019 for thumbs [10]

In figures, 2 and 3, γ is a parameter. If γ = 0, sampling is done uniformly; if γ > 0, sampling is biased towards the items with higher score. Thus, γ should be chosen in the range [0, 1] and for γ > 1 the bias is large.

In their paper [10], the user provides the feedback about preference - that is, two items are shown to the user and he is allowed to choose one of them indicating his preference. In our model, we implicitly assume a user's preference. For instance, if we are comparing two businesses that the user has reviewed, we first look at the rating and assume that the user prefers the one with the higher rating. However, if the rating for both businesses is equal, then we do a sentiment analysis on the review text using Alchemy API [4] to decide the preference. Implicit comparison may not be reasonable all the time, as

we may be comparing businesses of different genre. For example, can we compare a Thai restaurant with an Indian restaurant? The user may have a different scale of rating for Indian restaurants as opposed to Thai, or the user might have high expectations from an Indian cuisine and rate it lower and might have visited a Thai cuisine for the first time and rated it higher. The assumption we make is that the user uniformly rates all kinds on businesses on a similar scale of liking up to a certain extent. In addition, we use a method called Batching (Section 4.4) that mitigates the chance of comparing dissimilar businesses.

3 Yelp dataset

Yelp is a startup based in San Francisco, CA founded in 2004 to connect people with local businesses (restaurants, hair salons, dentists etc.). Very often, people “Yelp” the place they are about to visit, or take Yelp’s help to find out what businesses are doing a great job and what businesses are not. Yelp is a platform where users can create content, in the form of reviews, check-ins, ratings etc. about any business they like or dislike. Yelp has an average of 142 million monthly unique visitors in Q1 of 2015, and Yelpers have written over 77 million local reviews [1].

Yelp has been hosting its own Dataset challenge called “Yelp Dataset Challenge” [5] where they encourage students to use the dataset they exposed to come up with research ideas that might help Yelp. Their current dataset is large and rich, and includes information about the local businesses, reviews and users from 10 cities across 4 countries [5] mapped in Figure 4 below.

Figure 4: Overview of the cities involved in the Yelp Dataset [5]

The Yelp dataset is exposed for the following cities: Edinburgh (U.K), Karlsruhe (Germany), Montreal and Waterloo (Canada), Pittsburgh, Charlotte, Urbana-Champaign, Phoenix, Las Vegas, Madison (U.S).

Before getting into the details of the dataset, let us understand some Yelp vocabulary. A *business* - can be any public or private organization that requires public interest, and a place people often visit – it can be a bar, a restaurant, gym, hospital, theatre, and spa and so on. In addition, each of these kinds can have sub categories, for example, a restaurant maybe Indian, American, Thai, and Chinese and so on. A *User* - in Yelp's context, is any person registered with the Yelp's service (website) and is able to write reviews, tips and rate businesses. A *Review* - can be any user's opinion about a business (or an entity of the business) in form of

text and more likely than not, the user will have visited and experienced the offerings of a business before reviewing it. A *Tip* - is a way to pass along some key information about a business, such as the best time to go or favorite dish, without writing a full review about user's experiences. Lastly, a *Check-In* – is an event raised by a user, signifying his presence at a particular location (business).

The Yelp dataset has *1.6M* reviews and *500K* tips given by *366K* users for *61K* businesses. It comes with *481K* business attributes such as hours, parking availability, ambience etc. Yelp also has a social network of user's (which is of most importance to us), and this social network of *366K* users form a total of *2.9M* social edges [5].

In this thesis, we make use of the User, Review and Business entities of the dataset. Each of these dataset entities have been presented to us in a separate JSON file, and each line in the file relates to a single object of User, Review or Business respectively. If we look more closely, a single JSON object for each of these entities looks like this:

Business [5]

```
{
 'type': 'business',
 'business_id': (a unique identifier for this business),
 'name': (the full business name),
 'neighborhoods': (a list of neighborhood names, might be empty),
```

```
 'full_address': (localized address),
 'city': (city),
 'state': (state),
 'latitude': (latitude),
 'longitude': (longitude),
 'stars': (star rating, rounded to half-stars),
 'review_count': (review count),
 'photo_url': (photo url),
 'categories': [(localized category names)]
 'open': (is the business still open for business?),
 'schools': (nearby universities),
 'url': (yelp url)
}
```

User [5]

```
{
 'type': 'user',
 'user_id': (unique user identifier),
 'name': (first name, last initial, like 'Matt J.'),
 'review_count': (review count),
 'average_stars': (floating point average, like 4.31),
 'votes': {
   'useful': (count of useful votes across all reviews),
   'funny': (count of funny votes across all reviews),
   'cool': (count of cool votes across all reviews)
 }
 'friends' : [array of friend ids],
}
```

Review [5]

```
{
```

```
 'type': 'review',
 'business_id': (the identifier of the reviewed business),
 'user_id': (the identifier of the authoring user),
 'stars': (star rating, integer 1-5),
 'text': (review text),
 'date': (date, formatted like '2011-04-19'),
 'votes': {
   'useful': (count of useful votes),
   'funny': (count of funny votes),
   'cool': (count of cool votes)
 }
}
```

If we look at it from a database perspective - business_id, user_id form our primary keys for Business and User entities, and foreign keys for our Review entity, and this relation helps us figure that “User U wrote review R for business B”. Moreover, the User entity also has an array of friend user_ids, forming a connected social network of friends.

4 Ranking businesses using preference pair aggregation

4.1 Preference pairs

What is preference? Preference comes into context when two or more things are being compared to each other. If a person likes restaurant A better than restaurant B, that means he prefers A to B. Using this simple analogy, we define a preference pair P {U, A, B} to be – User U likes entity A better than entity B. In the past, many papers concentrated on using preferences of the user as the basis of ranking objects [10] [11]. The easiest and most reliable way of knowing the preference of a user is to take a direct feedback. That is, to present the user with two items and ask for his opinion as to which one he likes better, and the paper by Anish, Atish and Srinivas [10] follows this pattern. This would result in a reliable outcome, as there are no assumptions about user's preference.

However, can we infer the preference of a user based on his indirect feedback? Can we say that a user likes a product A better than product B if he rates A 4 stars and B 3 stars? The answer is probably a partial Yes (we will explain this in section 4.3.2). What if the user rates both A and

B 4 stars (the same)? Can we look at his review text to analyze his sentiment? Maybe yes and this exactly what we try to do in our model.

If a user U rates a business B1 higher than a business B2, we infer that U likes B1 better than B2. Moreover, if he rates both the businesses the same, which is a common case, we look at the review text for each of the business reviews and perform a sentiment analysis on it to decide the preference. Therefore, in our graph of businesses (nodes), a directed link indicates a user's preference.

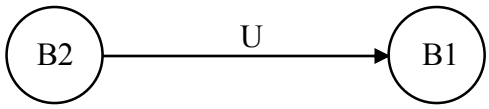

Figure 5: Preference - indicating user U prefers business B1 to B2

Figure 5 shows the preference of a user U about two businesses B1 and B2, indicating U prefers B1 to B2. In our graph, the preference is indicated by the direction of the edge; an edge direction into a node indicates that it is more preferred.

4.2 Forming the graph

For the user of interest, we gather all the preference pairs based on the businesses he rated and reviewed. The way we do it is, we look at every

review the user has written and compare it with every other review, and form preference pairs for the businesses he reviewed. As an example, if a user U has written three reviews for businesses {A, B, C}, then we perform the following comparisons: {A, B}, {B, C}, {A, C}, and we get three preference pairs. For n reviews the total number of preference pairs will be nC2 (from the formula nCr); mathematically – it is just finding all possible combinations of size 2 from a set of n reviews. This will give us a directed graph, where each node indicates a business and each directed link indicates the preference of the user. We consider users who have not reviewed or who just reviewed one business as outliers.

For the user of interest, once we gather all his preference pairs, we look for his friends within Yelp and gather their preference pairs too. All of the resulting nodes for a connected circle of users will form our graph.

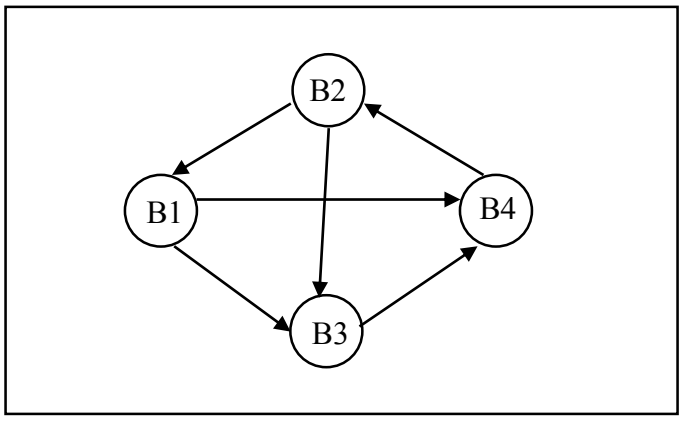

Figure 6: A simple preference pair graph for a single user

Figure 6 shows a simple preference graph for a single user. We see a link between every pair of nodes indicative of a preference comparison. Moreover, the total number of edges will be nC2 where n is the number of nodes.

4.3 Assumptions and inconsistencies

4.3.1 Outliers

For our model, any user who has not been active on Yelp is pretty much an outlier. We decide on the activeness factor based on the total number of reviews ever written by the user. If the user wrote one or no reviews, he/she would be an outlier. The reason being, unless the user has a minimum of two reviews, there cannot be a preference pair. In addition, a user is considered an outlier if he no friends. The main idea behind our thesis is to discover and recommend businesses that are liked by the user's friends. Hence, if the user's friend circle is not concrete enough, we believe that the results would be dissatisfactory. Furthermore, we also look at the cumulative businesses reviewed by a user's circle. More often than not, cumulative businesses reviewed are greater than two. However, there are cases where even if a user has a

few friends there might be a chance that none of them reviewed any of the businesses yet. This forms an obvious outlier for our model.

4.3.2 Business dissimilarities

How do we decide if two businesses are comparable? Can we compare any two random businesses based on the rating or review text? Can we compare a Spa to a Restaurant?

We only compare two businesses if they are, in some way, similar to each other. We think that comparing a Spa to a restaurant will not yield the right intention of user's preference because the metric for quality of a Spa for any given user can be different from the metric for quality of a restaurant. For example, when a user visits a spa he might rate the business based on the service and the masseuse; whereas the same user might rate a restaurant based on the food and the ambience. We use a concept called Batching (section 4.3.5) to avoid the problem of comparing dissimilar business.

4.3.3 Cycles for one user only graph

A one user only graph is a special case where an active user does not have any friend (which is a rare scenario), and this is a possible outlier for our model too, because we will be potentially recommending him

businesses that he already visited. However, we will look at how cycles are not possible in this case.

Intrinsically, when we compare any two businesses there is always a winner. If a user U has no friends, and has visited exactly two restaurants A and B, there is just going to be one comparison between A and B, and we will have a winner, forming a single link between A and B in our preference pair graph.

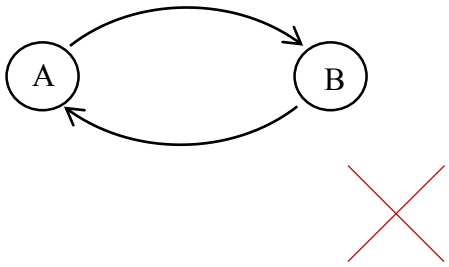

Figure 7: Cycle not possible for a single user only graph with two businesses

Let us take an example where the user visited three restaurants A, B and C. For our model, is it possible for a cycle formation for a single user scenario? Let us say - A, B and C all have different ratings and we will have a winner just by comparing them. In addition, there would only be three comparisons {A, B}, {B, C} and {C, A}. Let us say A > B and B > C; for a cycle to form C > A has to be true (we use > to indicate

preference), but transitively A > C. Hence, it would not be possible to form a cycle (directed) with three businesses for our model in a single user case.

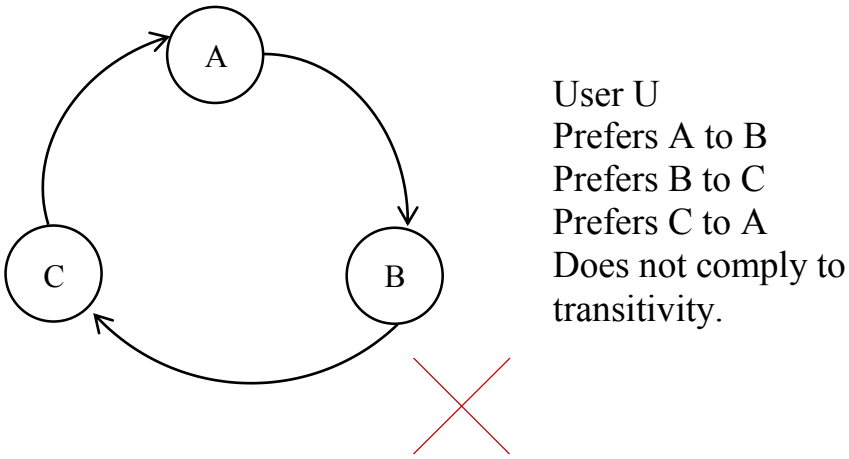

Figure 8: Cycle not possible for a single user only graph with three businesses

4.3.4 Cycles for a user circle graph

The cases discussed in the section above [4.3.3] are possible for a multi-user graph. Each user will have his/her own preference pair links, and there is every chance of a cycle forming. For example, let us assume that user U1 reviewed two businesses {A, B} and he preferred A to B, and user U2 (friend on U1) also visited {A, B} and he preferred B to A. Therefore, when we aggregate the graphs of these two users, we get a cycle as in Figure 7. We do not explicitly handle the cycles in our model, but we rely on the Zermelo, PageRank and In-degree ranking algorithms' intrinsic behavior of handling cycles.

4.4 Batching

Batching is a simple process of grouping similar businesses together before comparing. The reason we want to do this is to avoid comparing two dissimilar businesses, like comparing a Spa to a Thai restaurant. For a user, the metric for quality for a spa can be different from the metric for quality for a restaurant when is writes reviews. Therefore, it would be unfair to compare two dissimilar businesses.

Yelp's dataset can comprise of a varied set of categories ranging from medical health to restaurants to entertainment and so on [6]. Moreover, each of those categories might have sub-categories. For example, a restaurant is a category, and it in-turn can have a list of sub-categories like American, Indian, Chinese etc. So at what level in this tree of hierarchical categories do we operate? We choose to operate at level one, which strikes a balance between a good grouping and size of the group. For example, restaurants would be at level one of the category tree, and this will include all kinds of restaurants and cuisines and it gives us an opportunity to act on a large dataset. As we go deeper into the category tree, the dataset gets divided across various categories, and will leave us a small dataset to act upon.

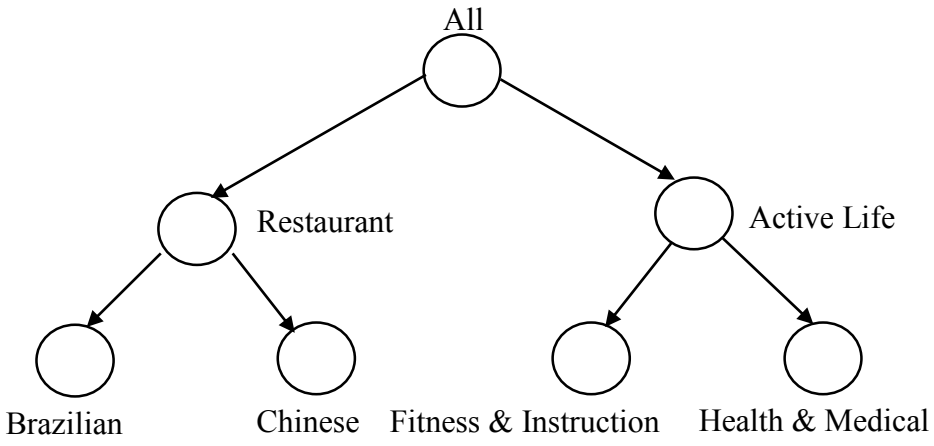

Figure 9: An example of category tree for Yelp with two levels

This way we try to mitigate unfair business comparisons; but a caveat of this approach is that - the recommendations that will be presented to any user will be categorized into different level-one categories of businesses.

4.5 The idea

There have been many ranking mechanisms in the past and one of the popular ones is PageRank. PageRank [12] [13] uses the link structure of the Internet to rank webpages. The idea behind PageRank is that the quality of the page being ranked is determined by the quality of its backlinks. Then there is HITS [14] (Hyperspace Induces Topic Search) which forms hubs and authorities and ranks nodes based on that. These ranking mechanisms make a lot of sense for ranking webpages, as the quality of the webpages that lead to the webpage being examined is of

utmost importance. In-degree counting is another method, where we rank nodes of a graph directly based on the count of incoming edges of the node. Will these ranking mechanisms yield good results for our paired-preference model too? On the contrary, for our setup, the nodes in the graph are businesses visited by a circle of users and the links between them are their preferences. That is, if there is a directed link from business A to business B, it means that the user prefers B to A. Each link in our social graph is an indication of a personal preference for one particular user. In the basic setup we do not assign any weight to the nodes or the links in our model, as it completely relies on the in and out-degrees.

Figure 10 below explains the overview of our paired preference ranking model's architecture. For the user of interest, we gather all his friends, and for each of the friend we gather all his preferences using his/her reviewed businesses. This will form our social graph of paired preferences. Then we run a ranking mechanism to rank the businesses.

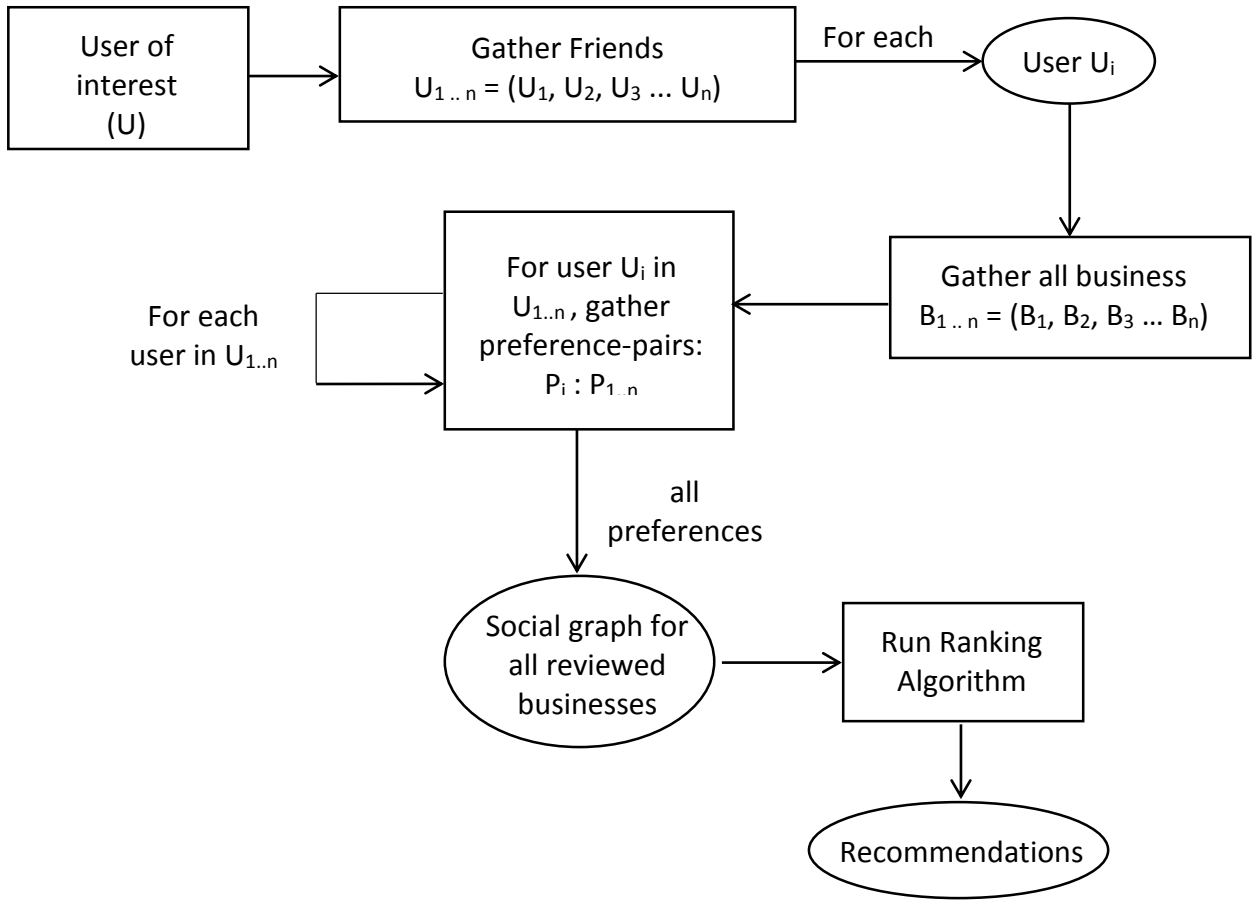

Figure 10: Architecture for our ranking mechanism

We try to make use of Zermelo's probability model, In-degree counting, and PageRank as our prime ranking mechanisms and try to analyze which one of these performs better for our setup.

Ernst Zermelo's probability model looks at each business as a player in a competition, assuming each game will result in only wins/ loses. The idea is to calculate the strength of each player (business) for our setup. In-degree counting ranks nodes of a graph directly based on the count of incoming edges of the node. PageRank [12] [13] uses the link

structure of the Internet to rank webpages; the quality of the page being ranked is determined by the quality of its backlinks.

The following subsections will give a quick run-down of the whole process, and a detailed explanation of Zermelo's theory, In-degree counting and PageRank, and try to analyze the best model for our setup.

4.5.1 Get user circle

The first step in our algorithm is to gather the list of friends for the user of interest. The idea behind considering the user's circle for recommendations is based on the assumption that a group of friends might have similar likes and dislikes. More often than not, in a real world scenario, a friend, or a friend of a friend or a person you know will have recommended most of the places you visit. Moreover, recommendations made by people you know are given higher priority than places suggested by random people.

Yelp is primarily a word of mouth review website, but it also has social framework to it. Every user can have a list of friends on Yelp (just like on Facebook), or if you do not know a person that well but you like his/her reviews, you can follow that person (just like on Twitter). Yelp's

dataset gives us a list of ids of friends for every user, and we use them to form the user's circle.

4.5.2 Get reviewed businesses

The next step is to gather the list businesses that the user and his circle of friends have reviewed and this list forms the basis of our model as the ranking will be performed only on these businesses. When a user writes a review on Yelp, there are two things he/she can do:

1. Rate the business on a scale of 1 to 5, with possible half stars
2. Write a text review.

The first gives us a quantitative measure of the user's affinity for that business and second gives us a qualitative measure expressed in the form of users' own words. We take into account both of these measures while forming preference pairs.

4.5.3 Get preference pairs

For every user U_i in the circle, we gather his/her paired preferences by performing pairwise comparison. We compare every business reviewed to every other business reviewed by a user. We do this for the entire user circle, one user at a time. The comparison is implicit, in the sense,

we do not acquire any user feedback about the preference, and rather we compare the businesses as a two-step approach:

1. In this step, we compare the businesses using the rating the user has given while reviewing them. It is a simple quantitative comparison. To ensure that the businesses being compared are similar we use a process called Batching [section 4.4]. The business with the higher rating value wins. Our model requires us to have a winner/loser for every comparison we make; but there is a good chance that the user might rate two businesses the same. In that case, we resort to step 2.
2. Along with a quantitative rating, the user also writes a textual comment while reviewing a business. In case of a tie in step 1, we perform a sentiment analysis (using Alchemy API [4]) on the review text for both the businesses being compared, and decide on a winner/loser based on the outcome. The Alchemy API takes a string as an input (we pass our review text), and returns a value between -1 and 1, indicating positive and negative sentiment of the text. A value closer to -1 exhibits a stronger negative sentiment, and the value closer to 1 exhibits a stronger positive sentiment. We use this value to decide a winner/loser for our comparison.

We do the above for every user in the user's circle. For each user U_i in the circle having n reviews the total number of preference pairs will be n_uC2 (from the formula nCr); mathematically – it is just finding all possible combinations of size 2 from a set of n reviews. Hence for a circle of U users there would be (U * n_uC2) set of preference pairs.

4.5.4 Social graph

Using the preference pairs above, we form our social graph, and we say social in the sense that we include preferences of the user along with preferences of his friends on Yelp. Each node in our graph is a business and the directed links from one business to another indicate user's preference. We run three different ranking algorithms (Zermelo's probability model, PageRank, In-degree counting) on our social graph to see which one yields better results.

4.5.5 Zermelo's probability model

Ernst Zermelo a renowned mathematician of early 19th century introduced a simple probability model for game outcomes as a function of player's unknown strengths by using the method of maximum likelihood estimation in 1928, specifically for chess tournaments. Later on, Bradley-Terry (1952) and Mosteller (1951) extensively explored the field of paired comparisons, and hence in literature, it is often referred

to as Bradley-Terry model due to a well-cited paper by those authors [15]. The research in paired comparisons has been active ever since. Later on in 1978, Arpad Elo modelled a rating system for rating chess players that was loosely based on Zermelo's idea [16] [17].

Zermelo's paper is concerned mostly with measuring relative strengths of players in an unbalanced game, that is, a game in which not every player competes against every other player the same number of times. Zermelo introduced a probability model to solve this as a function of players' unknown strengths. The model assumes that a game will always result in a win or a loss. As an analogy to our dataset, we think of every comparison we perform between two businesses (for a user) as a game and the strengths of businesses as its quality.

Letting B_p and B_q be competitors (businesses) p and q, the model in its simplistic form is given by [8]

$$P(\text{Bp } defeats\ Bq) = \frac{Sp}{Sp + Sq}$$

where S_p and S_q are unknown player strengths that are to be estimated. Computationally, it is more convenient to work with a re-parameterized version of this model using logarithms. Setting $V_p = \ln(S_p)$ for player B_p, the model can be re-written as [8]

$$P(Bp\ defeats\ Bq) = \frac{\exp(Vp)}{\exp(Vp) + \exp(Vq)} = \frac{1}{1 + \exp(Vq - Vp)}$$

The base of the logarithm is not of much importance, but the above expression is a standard logistic cumulative distribution function evaluated at (V_q – V_p). Though Zermelo considered logistic cumulative distribution as the de facto, Thurstone's model [18] [19] [20] on the other hand uses Gaussian cumulative distribution. Although these two models are conceptually different, it appears that in practice the difference is negligible [8].

The model requires the data to be fitted by a logistic regression. Not just that, but in the modern context where datasets are really large, logistic regression performs really well. Therefore, we need to find the values for (V_q - V_p) that best fits. We will use a maximum likelihood estimate to fit, which is a popular choice and the original estimate used in Zermelo's model.

Zermelo in his paper, for optimizing the likelihood function, decomposes the data set into disjoint sets of players. Each set is called a partial or prime tournament. The players can be decomposed into irreducible prime tournaments if, for a fixed prime tournament, every player not in the prime tournament has won all games, lost all games, or have not played a game with any player from the prime tournament.

Along these lines, Zermelo performs an ordering of partial tournaments such that for a j^{th} partial tournament C_j, every player in a partial ordering before C_j did not defeat a player in C_j. Moreover, every player in the partial ordering after C_j did not lose to any player in C_j [17]. Hence, according to Zermelo's paper, it is necessary that the dataset should be decomposable in a way that ordering exists.

In 1952, a paper by Ford [21], tackles the problem of ranking from binary comparisons, focusing on estimating strengths of players in irreducible datasets. Ford restates the condition for irreducibility as - If in every possible partition of players into two non-empty subsets, some player in the second subset has defeated a player in the first subset. Ford also demonstrates that this condition is sufficient for a unique optimum of Zermelo's likelihood function [17]. In other words, it is stating a strong connectivity condition for a graph.

Does our social graph satisfy the condition of irreducibility? Yes, mostly and most of the times, every node in our graph has at least one in or out links, and we can be sure of that by eliminating our outliers that are disconnected from our graph (see section 4.3.1 Outliers). Therefore, if we were to partition our graph into two parts (or two subsets), there would always be a link connecting the two subsets indicating irreducibility. Very rarely, it is possible that there might be a partition,

which is completely disconnected from the other, and even in that case, the disconnected partition usually has very few businesses, which we can safely ignore. We will check for graph's strong connectivity in the next section.

```
func runZermelo(List preferencePairs, List friends) {
     int[][] winMatrix = populateWinMatrix(preferencePairs)
     float[] gammaArray = new float[winMatrix.length]
     for i = 0 to gammaArray.length
          gammaArray[i] = 1 / winMatrix.length
     float epsilon = 0.01
     while (true) {
          gammaArray = EZUpdate(winMatrix, gammaArray)
          double newlikelihood = getLogLikelihood(winMatrix,
          gammaArray);
          if (Math.abs(newlikelihood - likelihood) < epsilon)
                    break
          likelihood = newlikelihood
     }
     return gammaArray;
}
```

Function 1: Overview of Zermelo's Algorithm [8]

The getLogLikelihood() function from the above snippet forms the gist of this model. From [8], we can explain the calculation of likelihood as follows: given the parameters gamma[i], the probability that a graph G will be realized is the product of each edge appearing with correct orientation as per binomial distribution, and this can be given by (product over pairs i, j):

$$P\left(\frac{G}{gamma[i]}\right) = W[i][j] + W[j][i]\ choose\ W[i][j] * \left(\frac{gamma[i]}{(gamma[i] + gamma[j])}\right)^{W[i][j]} * \left(\frac{gamma[j]}{(gamma[i] + gamma[j])}\right)^{W[j][i]}$$

The goal of Zermelo's model is to find gamma that maximizes this value. To simplify, in the above equation, the first term does not handle any gamma values, and hence can be ignored. We can further simplify the equation by taking log as it is a monotonically increasing function. Therefore, this simplifies the problem to maximizing the sum over all pairs (i, j):

$$P\left(\frac{G}{gamma[i]}\right) = W[i][j] * (\log(gamma[i]) - \log(gamma[i] + \log(gamma[j]))$$

After the algorithm converges, the final gamma values give us the ranking of all businesses reviewed.

For the purposes of programming we can represent the in and out degrees of all the nodes in our graph using a two-dimensional win

matrix. A two-dimensional win matrix for a user will look like Figure 11. We would need this step only while we run our model using Zermelo's probability model.

$$\begin{bmatrix} 0 & 1 & 20 \\ 5 & 0 & 1 \\ 2 & 3 & 0 \end{bmatrix}$$

Figure 11: Example of a win matrix M for three businesses

In the matrix M (Figure 11), M[i][j] is potentially the number of arcs from business i to business j, and at a low level each arc is considered as a win by j over i. From the above example we can say that business 3 had 20 wins over business 1 (from row 1 and column 3), and this indicates 20 arcs from business 1 to business 3. At this point, we might be wondering how can there even be more than one arc from business A to business B. It is possible in our case because we have accumulated preferences for various users into one graph and each arc will represent the preference of a different user.

4.5.6 In-degree counting

In graph theory, for a directed graph, the number of incoming edges gives the in-degree of a node. In-degree count is a simple method where the 'ability' of a node is directly proportional to the count of its in-degree edges. This ranking mechanism ranks the nodes based on this simple count – the higher the count, the better the rank. If two or more nodes have the same in-degree count, those nodes are ranked the same.

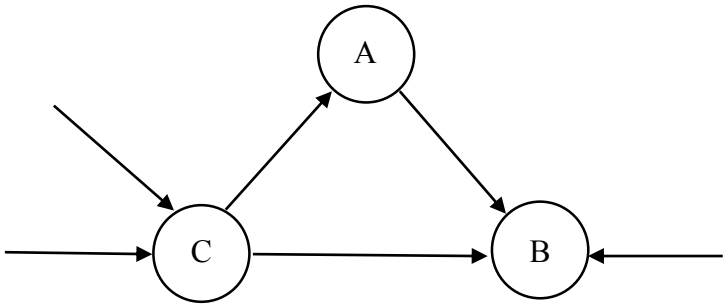

Fig **Figure 12: Example graph where A's in-degree is 1, B's in-degree is 3 and C's in-degree is 2**

ca

is 1, B's in-degree is 3 and C's in-degree is 2, and in-degree counting would rank {A, B, C} as B, C, A in that order.

4.5.7 PageRank

PageRank (or famously called Google PageRank) in simple words can be defined as the importance of a webpage (node) in the internet of billions of webpages. Importance of a page is determined mainly by its backlinks. For instance, if page A has a directed link to page B, we can

say that B has one backlink to A. In simple terms, PageRank can be given as:

$$PageRank\ of\ a\ Node\ (Site) = \sum \frac{PageRank\ of\ inbound\ link}{Number\ of\ links\ on\ that\ page}$$

PageRank is a vote, by all other webpages on the web, about how important a page is. A link to a page here is counted as a vote of support. The original paper for PageRank [13] defines PageRank as:

"We assume page A has pages T1...Tn which point to it (i.e., are citations). The parameter d is a damping factor, which can be set between 0 and 1. We usually set d to 0.85. There are more details about d in the next section. Also C(A) is defined as the number of links going out of page A. The PageRank of a page A is given as follows:

PR(A) = (1-d) + d (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))

Note that the PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one."

We use the equation above with a damping factor of d = 0.85 to rank our nodes in the preference graph, which will yield a ranking for the businesses reviewed.

We make use an open source graph/network framework called JUNG [22] (stands for Java Universal Network/Graph) – a software library that provides a common and extendible language for modeling, analysis, and visualization of data that can presented as a graph or network. JUNG includes implementations for a number of algorithms from graph theory and we make use of PageRank and estimate the scores of nodes in our social graph.

5 Results and analysis

5.1 Experimental setup

We have used Eclipse IDE for most of the development, and Java as our primary language to write algorithms. Yelp has exposed its dataset in the form of JSON. Mining the data from these files directly is a cumbersome and time-consuming task. Hence we perform some preprocessing over the dataset to create files f_u (where u runs from 1 to n), and u indicates a user, and n is the total number of user's Yelp dataset has to offer. Each file is named by a unique user id, and hence helps in easy file fetching. In addition, each file is going to have preference pair objects for that user in the following JSON format:


```
{
   "userid":"_977TriQp2lqEBv89FuNSA",
   "morepreferredbi":"SDwYQ6eSu1htn8vHWv128g",
   "lesspreferredbi":"OFBJtqWGSx6n6CchtVmoRw"
}
```


Figure 13: Preference pair JSON object for a user, where bi stands for business item

Figure 13 shows a JSON object for a preference pair of a user. *"userid"* indicates the user, *"morepreferredbi"* indicates the business that was

given more preference, and *"lesspreferredbi"* indicates the business that was given less preference.

So when we want to run our algorithm on a user and his friends, we just fetch the files from the local system using the user ids, and make use of the preference pairs directly.

Another important aspect in our model is sentiment analysis of review text. We make use of Alchemy API [4], a popular and reliable natural language processing framework, that takes a string as an input and gives a floating point value between -1 and 1 indicating positive, neutral or negative sentiment.

Moreover, we make use of JUNG [22] framework (Java Universal Network/Graph) to run PageRank over our social graph.

We make use of an open source framework Gephi [7] to perform graph analysis and draw business graphs as such. Gephi is a great tool that helped us in visualizing business graphs for Yelp users; and it has various functions exposed that came in very handy, like connected components, average degree, average path length etc., that gave great insights into the graph structure. We cannot import JSON directly into Gephi. Therefore, we had to write a function that converts our users' preferences into a CSV file that Gephi can understand. A sample CSV for

a user who reviewed three different businesses, where a single row can be interpreted as an arc from source to target by the user label is given below.

	A
1	Source;Target;Label
2	Lefty's Pizza;Cadillac Ranch;James
3	Lefty's Pizza;Prima Pizza & Cafe;James
4	Prima Pizza & Cafe;Cadillac Ranch;James

Figure 14: Sample CSV for a user who reviewed three businesses that Gephi understands

We have run our model on the first level category – restaurants.

5.2 Irreducibility

For optimizing the likelihood function, Zermelo's algorithm recommends the dataset to be irreducible. Irreducibility means if in every possible partition of businesses into two non-empty subsets, some business in the second subset has been preferred over another business in the first subset. In a way, this is testing for strong connectivity of the graph. We have picked a random sample of ten users from the Yelp dataset, and checked for strong connectivity in their social graph of businesses. The graph below is a forced-directed

graph drawing using Fruchterman Reingold algorithm; a class of algorithms that present graphs in an aesthetically pleasing way reducing the number of crossing edges in presentation.

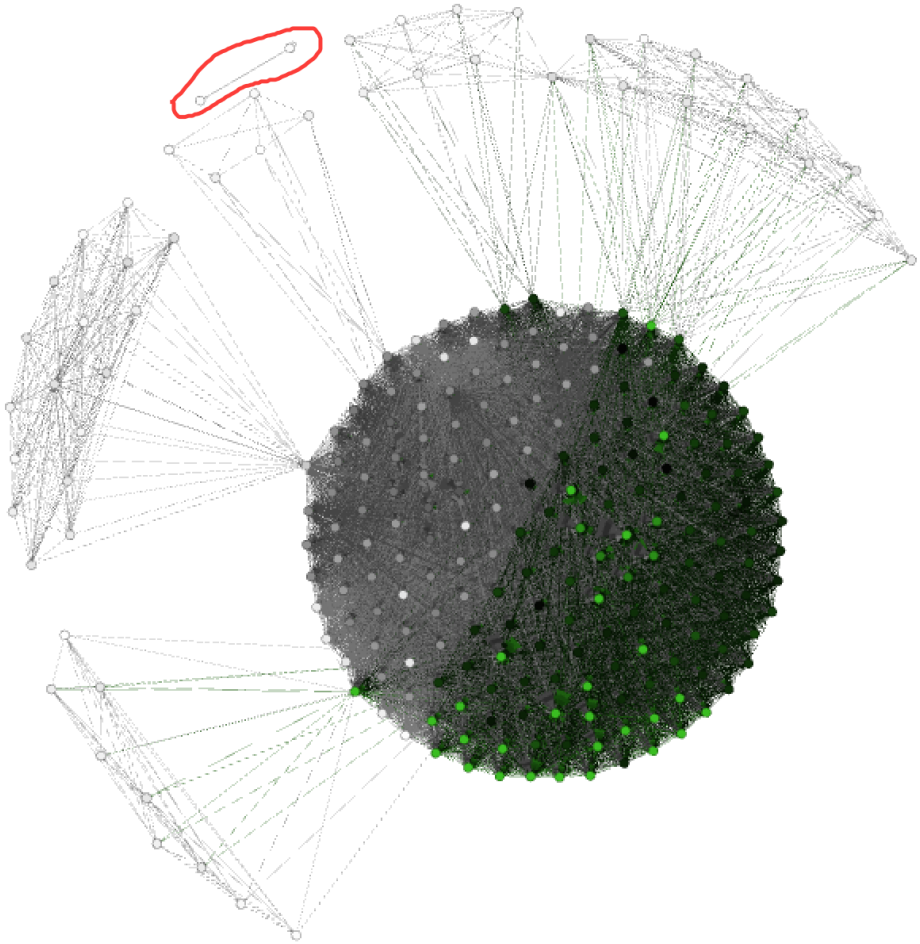

Figure 15: Social graph for a random user from Yelp dataset containing a disconnected hub generated using Gephi

In the experiment for strong connectivity of the graphs, for ten randomly selected users, 1428 nodes were examined out of which 76 nodes have exhibited disconnection from the graph. That is, if we were to partition the whole graph into a two partitions of 76 and 1428 of the

above resulted nodes, there would not be a link between the two partitions. Percentage wise it is 5.3% average disconnection. The 76 nodes that are not strongly connected to the graph were in the form of small hubs (the red area in figure 15), where each hub represents a single user's preference hub. Therefore, the total number of such hubs for those 76 disconnected nodes was 13, indicating 13 user preference hubs. Moreover, the total number of users involved in this experiment was 206 (of different smaller circle sizes). Mathematically, **6.3%** of the users, on an average, do not share any similarity while choosing businesses (restaurants) amongst their circle. On the flip side, 93.7% of the users, on an average, share some kind of similar interests with restaurants.

5.3 Recommendations via ranking businesses

We have generated recommendations using Zermelo's probability model, PageRank and in-degree counting in aggregation with paired preferences to compare against each another. Recommendations in our context are businesses that we think the user might visit/review in the future (mostly near future). For our setup, recommendations can be the top 10 or top 20 businesses of the ranking results (Zermelo or PageRank or In-degree). Typically, the recommendations generated by our model are always a subset of businesses visited/reviewed by the

user's friend circle. Therefore, if a user and his friend circle have reviewed only two restaurants during their lifetime in the Yelp network, we can present utmost two recommendations to the user of interest. Below is a sample output for ranking (both Zermelo and PageRank) showing only the top 10 ranked businesses for a user by name Mat. Circle size indicates total number of friends Mat has on Yelp network, and cumulative number of businesses reviewed indicate the sum of all businesses reviewed by each of Mat's friends. We can observe that there are a few common businesses from both the ranking systems in top ten, like Thai Basil, Grimaldi's Pizzeria, Cornish Pasta Company and so on, with just the ranks being different.

```
-----------------------Wgvq9BZwHLGY4jBqvJa8_g-----------------------
Name: Mat
Cumulative Preference Pairs gathered.
Circle size: 9
Cumulative number of businesses reviewed: 362
----------------- Zermelo -----------------
Initial Gamma: 0.002762431
Start likelihood : -244.57183265709318
End likelihood : -38.08623199767016
1. Durant's (cN6aBxe2mQvrQlzk26LyRQ)
2. Top Shelf Mexican Cantina (iJVZqAGZok2JNID_zGjFpQ)
3. Pizzeria Bianco (VVeogjZya58oiTxK7qUjAQ)
4. Thai Basil (0udEgNqy5rLR5pZ4kD19Og)
5. Four Peaks Brewing Co (JokKtdXU7zXHcr20Lrk29A)
6. Cornish Pasty Company (WNy1uzcmm_UHmTyR--o5IA)
7. Grimaldi's Pizzeria (dcd3C1gWv-vVdQ9XYV8Ubw)
8. Casey Moore's Oyster House (2ceeU8e3nZjaPfGmLwh4kg)
9. Dubliner (noLH_u4MJzfXYYHqcByjnA)
10. Eddie V's Prime Seafood (ZhMlXLXuZf5z7lxunHk2ww)
----------------- Page Rank -----------------
Damping factor d : 0.85
1. Grimaldi's (x4L42igQPv4TFlqGR2Wthg)
2. Four Peaks Brewing Co (JokKtdXU7zXHcr20Lrk29A)
3. Cornish Pasty Company (WNy1uzcmm_UHmTyR--o5IA)
4. Thai Basil (0udEgNqy5rLR5pZ4kD19Og)
5. Pizzeria Bianco (VVeogjZya58oiTxK7qUjAQ)
6. Rainbow Donuts Bakery and Cafe (ntH7Mo1JZdkglMiQXZy3Bw)
7. Pei Wei Asian Diner (gFGldPLVQdqbHQodU5TeUw)
8. Casey Moore's Oyster House (2ceeU8e3nZjaPfGmLwh4kg)
9. Grady's (Pju_pKfvEnUYV9wMVNsacg)
10. Chicago Gyros (-ftQeUsqwDkExRg6IYrubQ)
```

Figure 16: Sample output of recommendations using both Zermelo and PageRank systems for one user

5.4 Model comparison

We compare to see which among Zermelo, PageRank and In-degree yields better results in aggregation with paired preferences. In addition, we compare paired preference model to the rating sort.

The setup for comparison is slightly different. We split our dataset of reviews into two parts “before” and “after”, and we follow the popular 70:30 split, but in our case we are not really splitting into training and test datasets. Rather we will split our data as “before” dataset which we will use to generate recommendations using our model, and “after” dataset which we will use to see if the user has visited any of the ranked businesses. In the real world Yelp scenario, Yelp always notifies a user if his/her friends reviewed a business, and there is a section on their website to show the recent activity of all the friends. We assume that, somewhere between before and after split, the user might have been influenced by the activity of his friends, which is a very common scenario.

We have performed a few initial runs to get the 70:30 split for the review dataset and we figure that “*2013-07-31*” date splits the dataset in a ratio close to 70:30. That means all the businesses reviewed before this date land in “before” dataset and all the businesses reviewed after this date land in “after” dataset. Total number of reviews exposed in the Yelp dataset is *706646*; before dataset split has *496787* reviews; and after dataset split has *224281* reviews.

We pick a random set of 1000 users from the user data set, and run our preference paired model over the “before” dataset to get the ranking

(using Zermelo, PageRank and In-degree). Then each for user we check if he had visited/reviewed any of the ranked businesses (recommendations) from “before” dataset in the “after” dataset. We call this commonality; it simply means the number of businesses visited by the user in “after” dataset, that were reviewed by him or his friends in the before dataset.

Our comparison will be based on two evaluation metrics:

1. Mean Average Precision (MAP)
2. Number of wins

Our first measure is a popular statistical evaluation metric used in cases where there are multiple ordered correct answers in a list. The term “correct answers” is the same as commonality in our setup. We say we have a correct answer if one of the items from the recommendations (all) using before dataset, is also present in after dataset for the user of interest and we can have multiple correct answers. Formula for MAP is below:

$$\sum_{i=1}^{x} precision\ at\ i * change\ in\ recall\ at\ i$$

where precision at i is the percentage of correct items among first i recommendations; change is recall at i is 1/x if item at i is correct (for every correct item), otherwise zero; x is the number of recommendations we will present to the user. For the purposes of the experiment we do not fixate x to a value rather we take x to be the total number of businesses being ranked.

Our second measure for ranking is “number of wins” (not to be confused with win matrix from Figure. 11) which is a simple integer count. We assign a win to a ranking mechanism if it ranks a business that exhibits commonality better than other ranking mechanisms. From Figure 17 below, considering the first row and assuming the business id 7q1FpSXbE6XtLNg518pxDA exhibits commonality, PageRank ranked the business better at 108 than Zermelo at 110, so we assign a win to PageRank.

Zermelo Rank	Page Rank	User Id	Business Id
110	108	7MO0EG6_Dkw9VEvEq8UB7Q	7q1FpSXbE6XtLNg518pxDA
7	10	xcluigPrsq7Z_7fSTkW9WQ	fSs0EdKmLJ7VULejtsi9CQ
153	159	n6cNiKNMZeMtN3uGKSg-cA	j_pce4pG9krrBeYwUni8Pg
148	160	n6cNiKNMZeMtN3uGKSg-cA	YrlfgzLj0DPStJ1ESAS_Qg
57	20	lwm2oWPqCeYm42D_wV2mag	ngNvH4sxnH9aMukTZ67R_Q
102	108	9YVJtEzVfjfNujAfy0ysHg	jOuERtVf7QePnK9ZcdH5XA
137	144	9YVJtEzVfjfNujAfy0ysHg	QbmcCE_cLq4WO8ZMKlmaLw
28	7	1xxeOwZO3Vw9rZxmV-Nfgw	4bEjOyTaDG24SY5TxsaUNQ
58	43	1xxeOwZO3Vw9rZxmV-Nfgw	BW1JCE3Qj4uIokg7kLUL7w
109	106	1xxeOwZO3Vw9rZxmV-Nfgw	3fV9sL5LklD9ZSeONKmjkw
111	111	620LvQ2KsrADqIJfLVwHA	QbmcCE_cLq4WO8ZMKlmaLw
168	173	pc3wRHZZd9PHpjytIpftXw	tcphcsPONrP82kUyvcELoA

Figure 17: Few of many rank comparisons for different users

As another example, in Figure 16, let us say we had a commonality of one, and the business that is common in before and after dataset is “Cornish Pasty Company”. Now we look at the ranking systems to see which one gave this business a better ranking; and in this case, it is PageRank as the business is ranked #3 and Zermelo ranked the business #6. Hence, PageRank is assigned a win. Number of wins is a count of total number of wins a system gets.

We have divided the 1000 random sample users into 10 batches, each batch consisting of 100 users, and measured the MAP and number of wins for Zermelo, PageRank, and In-degree.

The plots in figures 18 and 19 below represent a comparison between Zermelo's probability model, PageRank and In-degree in aggregation with paired preference model.

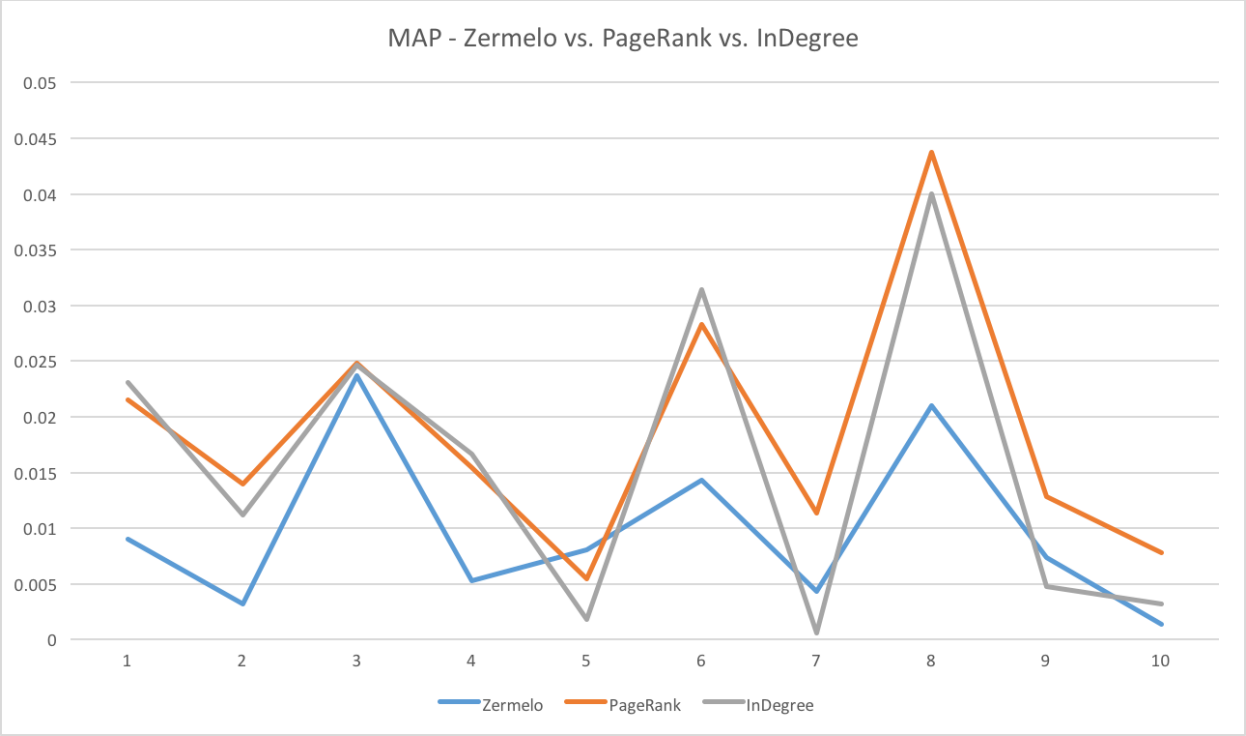

Figure 18: Mean Average Precision – Zermelo vs. PageRank vs. In-Degree (in aggregation with paired preference model)

In Figure 18, y-axis has the mean average precision values, and x-axis has 10 different samples of users, each sample consisting of 100 randomly picked users.

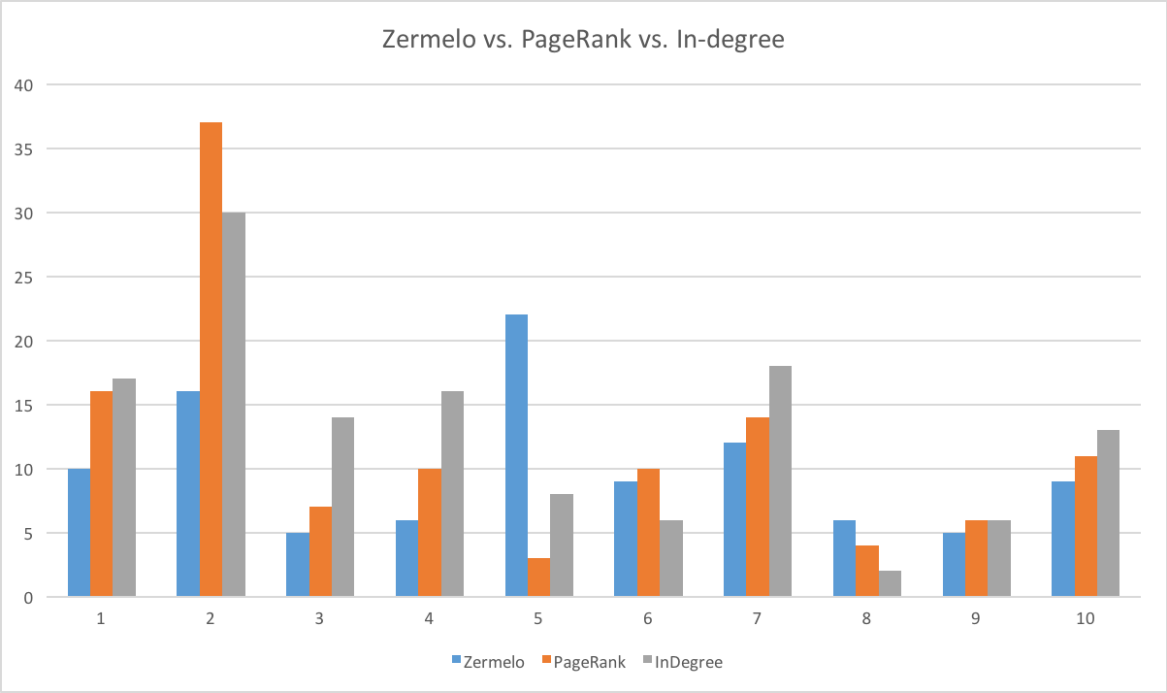

Figure 19: Number of wins – Zermelo vs. PageRank vs. In-Degree (in aggregation with paired preference model)

In Figure 19, y-axis has the number of wins' count, and x-axis has 10 different samples of users, each sample consisting of 100 randomly picked users.

We can clearly see that PageRank and In-degree in aggregation with paired preference model perform slightly better than Zermelo's probability model in aggregation with paired preference model.

Moreover, we compare our preference paired comparison model with the normal rating sort. Rating sort is a simple ranking system where we rank the businesses based on its rating (that is average user rating for that business). The way we rank businesses using rating sort for our

setup is as follows: We pick a random sample of 1000 users, and for each user we gather all of his friends. Then we fetch all the businesses reviewed by this set of users, and simply sort them based on the rating the business has. Then we compare these ranking results with paired preference model in aggregation with PageRank and In-Degree.

The plots in figures 20 and 21 below represent a comparison between In-Degree in aggregation with paired preference model and simple rating sort.

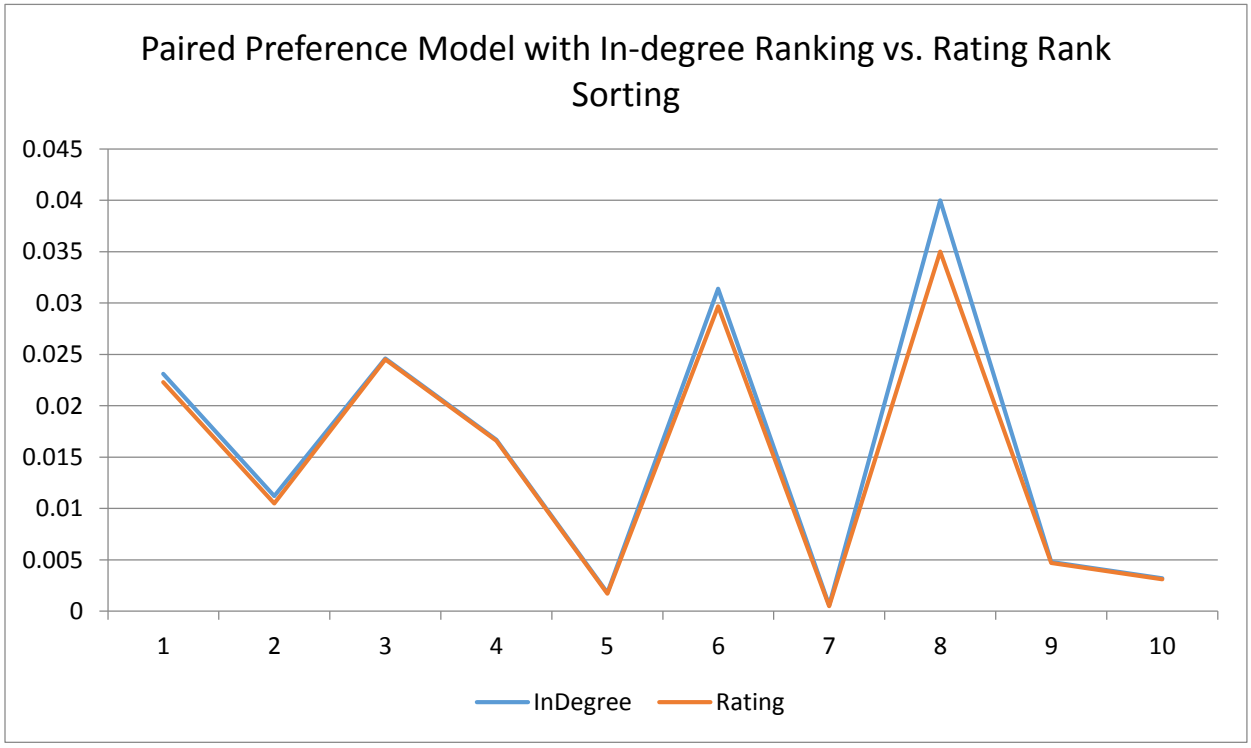

Figure 20: Mean Average Precision – Paired preference model + In-Degree vs. Rating sort

In Figure 20, y-axis has the mean average precision values, and x-axis has 10 different samples of users, each sample consisting of 100 randomly picked users.

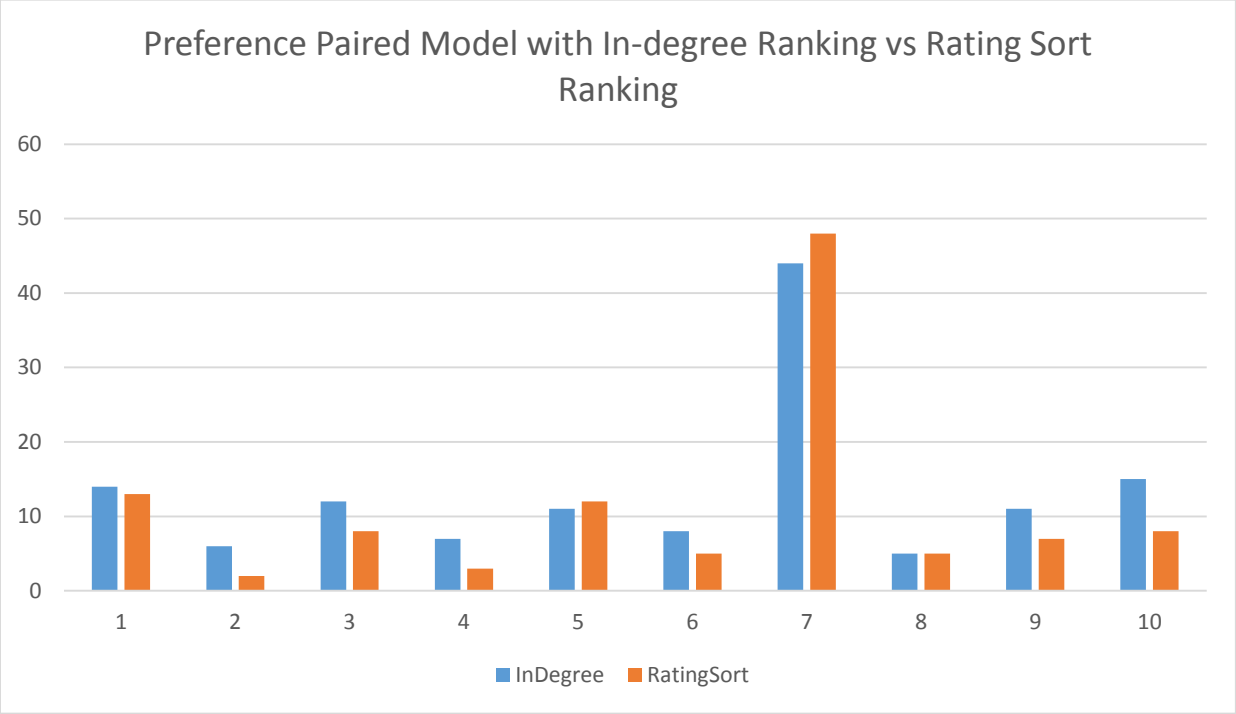

Figure 21: Number of wins – Paired preference model + In-Degree vs. Rating sort

In Figure 21, y-axis has the number of wins' count, and x-axis has 10 different samples of users, each sample consisting of 100 randomly picked users.

The plots in figures 22 and 23 below represent a comparison between PageRank in aggregation with paired preference model and simple rating sort.

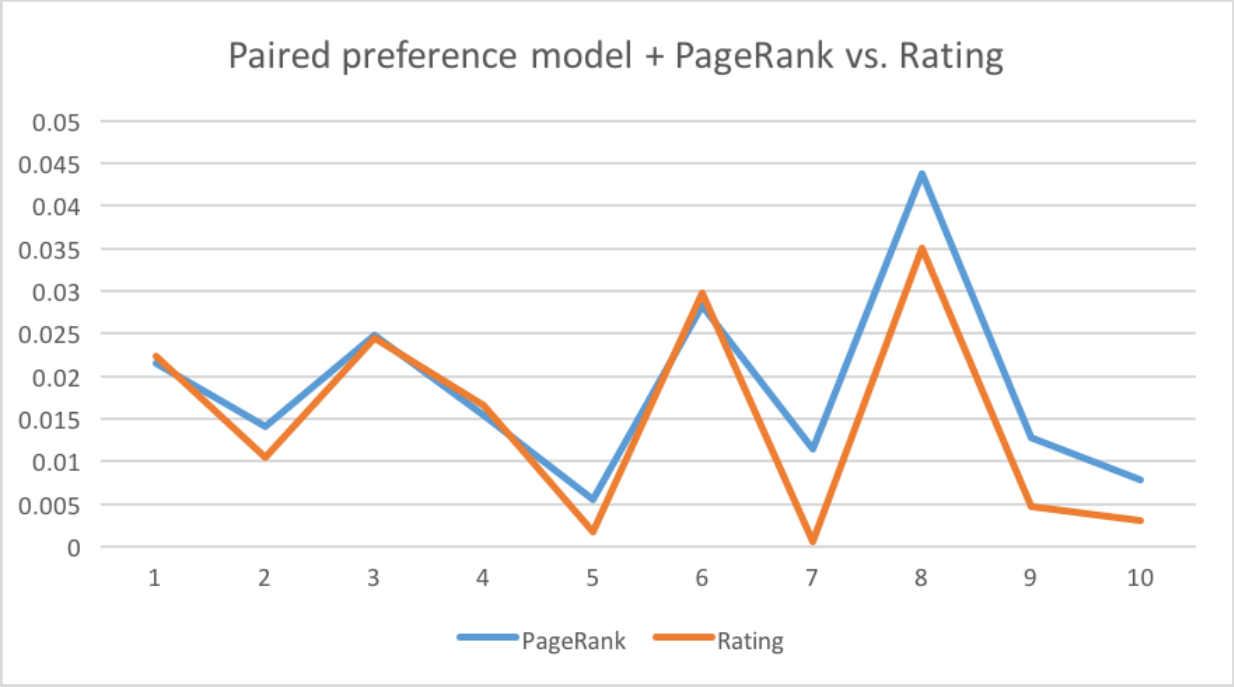

Figure 22: Mean Average Precision – Paired preference model + PageRank vs. Rating sort

In Figure 22, y-axis has the mean average precision values, and x-axis has 10 different samples of users, each sample consisting of 100 randomly picked users.

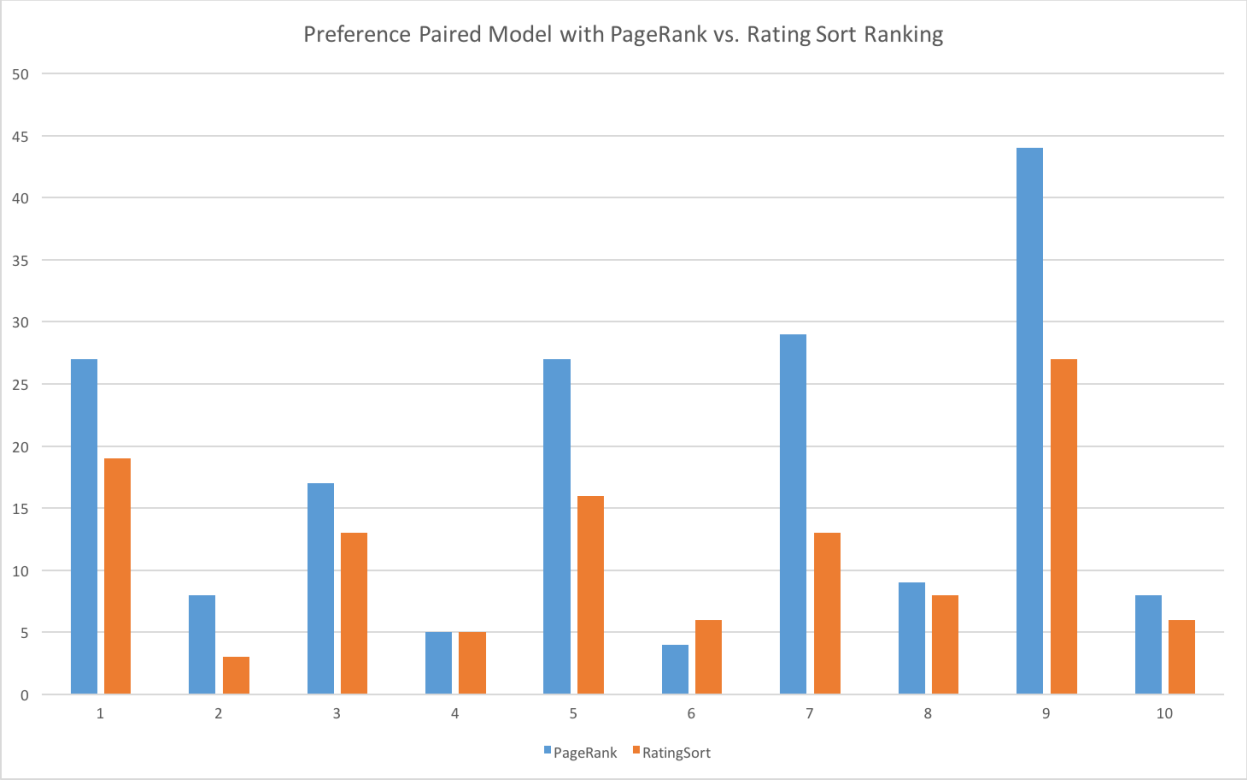

Figure 23: Number of wins – Paired preference model + PageRank vs. Rating sort

In Figure 23, y-axis has the number of wins' count, and x-axis has 10 different samples of users, each sample consisting of 100 randomly picked users.

From the graphs above, we can observe that PageRank and In-degree ranking work better in aggregation with paired preference model, with PageRank having a slight lead. The idea of how strong a node is based on its back-links and simple in-degree counting works surprisingly well for our model.

5.5 User behavior in Yelp using commonality

Commonality is the idea that a user visits a business in the “after” dataset, that has been recommended using the “before” dataset. In simple terms, commonality is indicating number of businesses visited by the users after a particular date d, which were suggested to him using the dataset from before the very date d. After we have split our dataset into two parts 70:30 based on the date “*2013-07-31*”, we try to make use the commonality to see how many of the users actually show interest in businesses visited by his friends in the past in the current Yelp setup.

Our experiment shows that, out of a random sample of 1000 users, 72 users exhibit commonality, inferring **7.2%** percentage of users visited restaurants that have been reviewed by his/her friends in the past.

6 Conclusion and future work

We have seen a different approach to provide recommendations for Yelp network based on the activity of a user's friend network and taking into account the implicit feedback given by the user about businesses in form of Yelp reviews. The main idea behind our approach was to use implicit feedback from the dataset to form preference pairs of businesses for a user, and aggregate those preference pairs to form our social business graph. Then we ran three different ranking mechanisms - Zermelo, PageRank and In-degree over our social graph to see which of these yield better results for our setup. A set of top businesses from this ranking will form our recommendations.

Though Zermelo's probability model is simple to understand and implement, we see that PageRank and In-degree perform a little better while ranking businesses, with PageRank having a slight lead. We also see that PageRank and In-degree in aggregation with paired preference model performs better than simple rating sort ranking.

We also unveil some interesting insights of the Yelp dataset like similarity of interests between users, percentage of users visiting businesses that were reviewed by their friends in the past etc.

Furthermore, many enhancements can be made to this model in the future to make it more efficient and robust. One of the major areas of improvement can be the number of comparisons made to form the social graph. Because we use implicit feedback to retrieve preference of a user to form preference pairs, we make nC2 comparisons for each user. So for a set of U users we make a total of U * n_uC2 comparisons, which are quite a lot. Reducing the number of comparisons performed per user will enhance the time complexity of the model.

In addition, we made use of Zermelo's probability model, PageRank and In-degree counting over the social graph to rank Yelp businesses. Exploring more of such ranking mechanisms that might fit well with our model would be great.

Furthermore, Yelp's dataset does not expose follower network information except the count (as of this date). Taking into account follower network along with friend network would boost the credibility of our model. Moreover, we make use of rating and review text to retrieve implicit feedback. Introducing more such influencing factors would improve the reliability of our model.

References

[1] Yelp statistics, *http://www.yelp.com/about* (Accessed June 17 2015)

[2] Yelp recommended reviews, *http://officialblog.yelp.com/2013/11/yelp-recommended-reviews.html* (Accessed June 17 2015)

[3] Yelp dataset, *https://www.yelp.com/academic_dataset* (Accessed June 17 2015)

[4] Alchemy API, *http://www.alchemyapi.com/api* (Accesses June 17 2015)

[5] Yelp dataset challenge, *http://www.yelp.com/dataset_challenge* (Accessed June 18 2015)

[6] Yelp categories, *https://www.yelp.com/developers/documentation/v2/all_category_list* (Accessed June 18 2015)

[7] Gephi, *http://gephi.github.io/* (Accessed June 18 2015)

[8] Zermelo's Approach, *https://compuzzle.wordpress.com/2013/02/06/from-zermelo-to-zuckerberg-the-method-of-paired-comparisions/* (Accessed June 20 2015)

[9] Peilin Yang and Hui Fang (2012). An Exploration of Ranking-based Strategy for contextual suggestion. In *TREC 2012 proceedings*.

[10] Anish Das Sarma, Atish Das Sarma, Srinivas Gollapudi (2010). Ranking mechanisms in twitter-like forums. In *WSDM 2010, pp. 21 – 30*.

[11] Rajat Bhattacharjee, Ashish Goel (2006). Incentive based ranking mechanisms. In *First workshop on the economics of networked systems, June 2006*.

[12] Lawrence Page, Sergey Brin, Rajeev Motwani and Terry Winograd (1999). The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[13] Sergey Brin and Lawrence Page (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *7th International WWW Conference, 1998*.

[14] Jon M. Kleinberg (1999). Authoritative Sources in a Hyperlinked Environment. In *Journal of the ACM, Vol. 46, pp. 668 – 677*.

[15] Bradley, Ralph A. and Terry, Milton E. (1952). The rank analysis of incomplete block designs. I. The method of paired comparisons. *Biometrika Vol.39, No. ¾, pp. 324 – 345*.

[16] Arpad Elo. The Rating of Chessplayers, Past and Present. Arco Publications, 1978.

[17] Introductory note to 1928 by Mark E. Glickman. *Ernst Zermelo Collected Works/ Gesammelte Werke, Vol. 2, Band 2, pp. 616 – 671*.

[18] Thurstone, Louis L. (1927). A law of comparative judgment. *Psychological review, Vol. 34, Issue 4, pp. 273 – 286*.

[19] Thurstone, Louis L. (1927). Psychophysical analysis. *The American Journal of Psychology, 1927, Vol. 38, Issue 3, pp. 368 – 389*.

[20] Thurstone, Louis L. (1927). The method of paired comparisons for social values*. The journal of Abnormal and Social Psychology, 1927, Vol. 21, Issue 4, pp. 384 – 400*.

[21] L. R. Ford Jr. (1957). Solution of a Ranking Problem from Binary Comparisons*. The American Mathematical Monthly, Vol. 64, Issue 8, pp. 28 – 33.*

[22] Java Universal Graph/Network framework, *http://jung.sourceforge.net/* (Accessed June 25 2015)